Mississippi State UNIVERSITY

Center for Air Sea Technology

RE-ENGINEERING A RELATIONAL DATABASE SYSTEM TO PRODUCE A PROTOTYPE OBJECT-ORIENTED GEOPHYSICAL DATABASE SYSTEM

by
Julia Hodges, Shekar Ramanathan, and Susan Bridges

Technical Report 01-95

20 June 1995

Prepared for: Office of Naval Research (Code 1242)
Under Research Grant N00014-92-J-4109

TECHNICAL REPORT 01-95

RE-ENGINEERING A RELATIONAL DATABASE SYSTEM TO PRODUCE A PROTOTYPE OBJECT-ORIENTED GEOPHYSICAL DATABASE SYSTEM

by

Julia Hodges[1], Shekar Ramanatham[2], and Susan Bridges[3]

[1]Associate Professor, Mississippi State University, Department of Computer Science, Mississippi State, MS 39762-9637

[2]Graduate Research Assistant, Mississippi State University, Department of Computer Science, Mississippi State, MS 39762-9637

[3]Assistant Professor, Mississippi State University, Department of Computer Science, Mississippi State, MS 39762-9637

20 June 1995

This research was supported by the Department of the Navy, Office of the Chief of Naval Research, under Contract/Research Grant Number N00014-92-J-4109 with the MSU Center for Air Sea Technology. The information contained in this publication does not necessarily reflect the position of the policy of the United State Government. No official endorsement should be inferred.

TABLE OF CONTENTS

Re-Engineering a Relational Database System to Produce a Prototype Object-Oriented Geophysical Database System

Julia Hodges
Shekar Ramanathan
Susan Bridges

Abstract

This document provides a description of (1) the design and development of a prototype object-oriented geophysical database system, and (2) the development of a re-engineering process for mapping from an existing relational database to an object-oriented database. Although it is widely recognized that the object-oriented approach has many advantages over traditional relational technology for scientific databases, one cannot ignore the large investments that have been made in existing relational databases. The re-engineering process described in this document provides a systematic method for (1) mapping from an existing relational schema to an object-oriented schema and for (2) mapping the actual data from the relational database into an object-oriented database.

Introduction

It is now widely recognized that an object-oriented database paradigm is more suitable than the relational model for scientific, engineering, and geographic information systems (Bertino and Martino 1991; Bhargava 1992). There are already many relational database systems in use, however, and the owners of these systems are not likely to choose to throw away the existing databases in order to move to object-oriented database technology. Therefore there is a need for a re-engineering process that can map an existing relational schema onto an object-oriented schema in a systematic way, then map the actual data from the relational database into an object-oriented database. In this paper, we describe our work in defining such a re-engineering process. We also describe how we have used the portions of this process that we have already defined to build a prototype object-oriented geophysical database from an existing relational database.

We begin with a summary of the advantages for scientific applications offered by the object-oriented database paradigm. We then describe the different kinds of data to be stored in the prototype object-oriented database system. Next we discuss the design of the object-oriented schema for the portion of the database which has been the focus of most of the work done so far (i.e., the grid data). We also describe the re-engineering process that was used to load the grid data by mapping from the existing relational grid data to the object-oriented grid data. Then we describe the preliminary work that has been done in the development of an object-oriented representation for other types of geophysical data stored in the relational database system. Finally we provide a summary and conclusions.

Motivation for Object-Oriented Approach

Blaha, Premerlani, and Rumbaugh (1988) provided four criteria for judging the "merit" of a database design:

1. performance: Does the structure of the database promote the availability of the data?; can users quickly retrieve and update relevant data?;
2. integrity: To what extent does the database guarantee that *correct* data is stored? (the definition of "correct" depends on the application);
3. understandability: How coherent is the structure of the database to end users, other database architects, and the original designers after a period of time?;
4. extensibility: How easily can the database be extended to new applications without disrupting ongoing work?

Engineering and scientific applications may be characterized by the need for handling complex data. That is, such an application may be required to deal with real-world objects which are structurally complex. An acoustic image, for example consists of rectangular cells called *texels* (Reed and Hussong 1989). Each texel, in turn, consists of a number of pixels. Object-oriented database systems allow an image to be described as a single complex entity, whereas relational systems require that the image be decomposed (or, in relational terminology, normalized) into multiple tables, with the different tables describing the components of an image. The end result is that the semantics of the entity being modeled are scattered across multiple relations rather than being packaged into a single object. The object-oriented approach allows a user to access the complex object as a whole as well as to access individual components of the object. Thus, for complex data, the object-oriented model meets the first and third criteria given above better than the relational model does.

The referential integrity constraints that must be enforced in relational database systems are the result of certain limitations of the relational model:

- the requirement that complex entities must be decomposed into a collection of two-dimensional tables, and

- the fact that relationships in a relational database are represented implicitly (such as through the use of foreign keys) rather than explicitly.

Consider, for example, an application in which we wish to represent students' class schedules. A relational schema for this application is shown in Figure 1. The attributes that form the primary key for each relation are underlined.

STUDENT (ID, name, classification)
CLASS (course_number, course_name, time, place)
ENROLLED (ID, course_number)

Figure 1. Relational Schema for Classes

What are the referential integrity constraints with which we must be concerned in this application? First, we must be sure that we do not enroll a student in a class that is not being offered. Second, we must be sure that we do not enroll a non-existent student in a class. Thus, any time that we store a new tuple in the ENROLLED relation, we must make sure that the student ID number already exists in the STUDENT relation and the course number already exists in the CLASS relation. (There are other integrity constraints that we would want to maintain, such as ensuring that no student is enrolled in two different classes at the same time of day, but we will not consider those here.) If the relational DBMS that we are using does not provide a mechanism for defining referential integrity constraints, then we must rely on each application program to check properly for any integrity violations.

The entity-relationship (ER) model offers some advantages over the relational model for capturing the semantics (or meaning) of the data in a database. For example, the ER model, originally defined by Chen (1976), makes a distinction between entities and relationships. In an ER design for our previous example (Figure 1), STUDENT and CLASS would be entities, whereas ENROLLED would be a relationship. In the relational model, relationships are represented using either foreign keys (for one-to-many relationships) or as separate relations (for many-to-many relationships). Thus the distinction between an entity and a relationship is blurred in the relational model. In addition, the ER model makes the mappings between entities involved in a relationship (one-to-one, one-to-many, or many-to-many) explicit, whereas this information is merely implied in a relational design. There are no commercial DBMSs based on the ER model, however, so that the ER model is frequently used as a design tool, with the resulting design being transformed into a relational schema which is then implemented using a relational DBMS. The popularity of the ER model as a design tool is reflected in the use of its concepts in many database design tools and the fact that there has been an annual international conference on the ER approach for more than ten years (Elmasri and Navathe 1994).

Relational database management systems (DBMSs) are attractive for a number of reasons (Blaha, Premerlani, and Rumbaugh 1988). Relational tables provide a very simple way of representing data.. The relational model is theoretically sound and well understood. Unlike the ER model, it is supported by a number of commercial DBMSs. Yet it is the simplicity of the relational model that makes it unsuitable for many complex applications. The requirement that relational data must fit into a two-dimensional table is too restrictive for representing more complex data. The ER model provides a slightly more abstract representation than the relational model (e.g., relationships are expressed explicitly in the ER model). But it does not provide a "substructure for entities and relationships" (Blaha, Premerlani, and Rumbaugh 1988). For example, the ER model does not support entities which may have subcomponents that are also entities, each of which may have subcomponents, etc.

The object-oriented paradigm supports the modeling of complex data and interrelationships "in a natural way" (Bertino and Martino 1991). That is, the model supports the definition of objects which have a complex structure, groupings of objects into classes, and the arrangement of object classes into an inheritance hierarchy. Not only does the model support the structural definition of an object, but "also the modeling of object behaviors and dynamic constraints" (Bertino and Martino 1991). The traditional relational model has no mechanisms for defining complex objects as single entities, for explicitly defining relationships among entities, or for defining the behavior of an object as a part of the definition of the object.

Object-oriented database management systems support *schema evolution* (the changing of the definition of the database) because many scientific and engineering applications are dynamic and need this capability. Relational database management systems provide some limited schema evolution capabilities such as adding new relations and adding new attributes to existing relations, but they do not support changes as complex as those found in many object-oriented database management systems (Bertino and Martino 1991).

Object-oriented database management systems also support *versions* (different states of the same object). As with schema evolution, the support of versions has been dictated by the needs of the applications. Bertino and Martino (1991) have described the need for version support as being "inherent in applications that are exploratory and evolutionary." For example, consider the processing of hull-mounted wide-swath bathymetric sonar data being done at NAVOCEANO at the Stennis Space Center. The scientists working with the imagery data need to derive mosaics and ocean bottom classifications, validate bathymetry data, and correlate the acoustic data with other sensor data. The acoustic imagery data can complement bathymetry data in providing geologists and engineers with ocean bottom topography. The acoustic imagery data provides information about the "microroughness, texture, and backscatter properties of the ocean floor" (Lingsch and Robinson 1992). Scientists may use several different techniques to derive mosaics from image data. The mosaics derived by different mechanisms could be represented as different versions. This will support testing of new algorithms for processing the image data in order to determine which ones reveal the features needed to validate new models. The support for schema evolution and versions gives the object-oriented DBMSs the ability to meet the fourth criterion given above in a more powerful manner than relational DBMSs.

The choice of which database model to use is dependent on the characteristics of the application which the database system is intended to model. The relational model was designed to support traditional data processing applications such as inventory control and order processing. The analysis of large quantities of scientific data, however, requires capabilities not provided in the relational model. As described by DeSanti and Gomsi (1994), such applications "are very dynamic and their database schema is usually very complex," and they require "the ability to handle the creation and evolution of schema of arbitrary complexity without a lot of programmer intervention." This is exactly the type of application for which object-oriented database systems are most useful.

Scientists at the Mississippi State University Center for Air Sea Technology (CAST), which is located at the Stennis Space Center, expressed an interest in pursuing the possibility of using object-oriented technology for storing large quantities of geophysical data currently stored in a relational database. In 1993, we evaluated several object-oriented database management systems for CAST and recommended that they purchase ObjectStore (Ramanathan and Hodges 1994a). Since then, we have worked closely with the CAST scientists in the design of an object-oriented schema for the data and in the development of a windows-based user interface for the resulting database system.

Description of the Geophysical Data

To demonstrate the advantages of the object-oriented approach for a geophysical database, we designed and implemented an object-oriented database system that contains a portion of the data found in the NEONS database system. NEONS (Naval Environmental Operational Nowcasting System) is a comprehensive system that includes software and procedures to support the "(a)nalysis of environmental data" (Jurkevics 1992). The NEONS database is a relational database that consists of four realms: primary, associative, descriptive, and geographic. The primary realm contains environmental data of four types: image data, grid data, latitude-latitude-time (llt) data, and line data. The associative realm contains information about the primary data such as time coverage, storage format, resolution, and grid geometry. The descriptive realm contains "descriptions of satellites, sensors, channels, orbital elements, grid geometries, and projections." The geographic realm "contains time-invariant data about the earth's geography" such as "coastlines, rivers, political boundaries, topography, bathymetry, and land-surface type" (Jurkevics 1992).

All of the primary data in the NEONS database is stored as packed bitstreams in order to save space. This introduces additional overhead for the packing and unpacking of the data, but the NEONS database designers thought that the decreased data volume would improve the overall I/O performance (Jurkevics 1992). The image data is packed into bitstreams, then stored in external files. The other three types of primary data are packed into bitstreams and then stored in relational tables.

The image data consists of multi-band images and overlays. These images can be in either satellite or registered coordinates. The images can be of any size, number of bands, and resolution. The grid data consists of the output produced by atmospheric and oceanographic analytical models, user-defined products, and gridded climatology data. Llt data represents measurements or reports taken at a particular point (latitude and longitude) at a particular time. The llt data is the most varied of the different types of primary data. It consists of "conventional environmental reports, earth-located satellite scene stations, and point climatology data" (Jurkevics 1992). Line data consists of a series of point observations (i.e., llt data) along a curved line.

For our prototype system, which we call ObjNEONS, we began by focusing on the grid data in the primary realm. The primary grid data is actual data values for various parameters (such as salinity or sea-surface temperature) at different grid points. We designed and developed an object-oriented schema for the grid data, then loaded the grid database by writing routines to map from the NEONS grid data to the object-oriented grid database by writing routines to map from the NEONS grid data to the object-oriented grid data. We also designed and implemented a windows-based interface for the object-oriented grid database. We have done some preliminary work on the image data and llt data portions of the object-oriented database. Most of our discussion in this document will be about the grid data portion of the database, although we do provide a brief description of the work done on the image and llt data.

We decided to store the primary data in our prototype system without packing the data. Eventually, we would like to conduct some performance experiments to compare the NEONS relational database and our object-oriented database in terms of the amount of storage required, the response time required for various operations, and the I/O time required for various operations.

Design of the Object-Oriented Grid Database Schema

In the design of the ObjNEONS database, we wanted to "exploit all the features provided by the object-oriented model to provide a view that represents the real world as much as possible" (Ramanathan and Hodges 1994b). We first had to familiarize ourselves with the grid data and what it represented to the scientists. We did this by producing an ER diagram for the relational grid data. The process for producing an ER model from a relational schema is called **reverse engineering.** It is the reverse of a well-known process for producing a relational schema from an ER diagram (described by Elmasri and Navathe (1994)). We then augmented the resulting ER diagram with domain information in order to produce an object-oriented schema, a process called **forward engineering.** The entire process of mapping from relational to ER to object-oriented is called **re-engineering** the database. This process, which is described later in this document, is summarized in Figure 2.

Initially, we provided an object-oriented view of the existing relational database (Ramanathan 1994) and developed a windows-based interface that allowed access to the data through this view (Wu 1993). This provided the CAST scientists with the opportunity to try out an object-oriented approach before investing in a new DBMS. Following an evaluation of commercial object-oriented DBMSs (Ramanathan and Hodges 1994a), CAST purchased ObjectStore. We then began the implementation of the object-oriented grid database system, including a graphical user interface called *Grid Data Browser* (Koduri 1994).

The Re-Engineering Process: Mapping from Relational to Object-Oriented

Our primary motivation for re-engineering an existing relational database rather than developing an object-oriented database "from scratch" is to be able to make use of the existing relational database to populate the object-oriented database. A number of approaches to this process have been reported in the literature. Markowitz, in collaboration with Shoshani (1989) and Makowsky (1990), proposed a process for reverse engineering a relational database schema to an extended entity-relationship (EER) structure. This is accomplished by considering key and inclusion dependencies defined in the relational schema. However, their approach assumes that the relational schema is well-designed, something which is often not true in practice.

Chiang, Barron, and Storey (1994) have defined a knowledge-based approach to extracting EER structures from a relational database. Premerlani and Blaha (1994) have proposed an approach for extracting an object-oriented database schema from a relational schema. Their approach makes use of a number of tools, including OMTool (an editor that produces object diagrams using OMT, or object-modeling technique, notation), SQL, AWK scripts, and various other programs and macros. In addition, they make use of manual analysis of the data. Based on their case studies, they have concluded that "[a] purely mechanical approach to reverse engineering of databases does not consider the transformations designers often apply in moving from design to implementation." Thus they think that the best approach is to use a "flexible, interactive approach" that provides the designer with "a suite of flexible, loosely coupled tools."

As with the other approaches, our re-engineering process is concerned in part with mapping from a relational schema to a schema represented by some other model. We have chosen to map to an object-oriented database schema for the reasons provided in an earlier section. Unlike other approaches that have been described in the literature, however, we have not limited our re-engineering process to schema transformation. We have taken a more extensive view of the process by including the mapping of the actual data from a relational database to an object-oriented database.

We shall first discuss the process of mapping from a relational schema to an object-oriented schema. After we have described the schema mapping process, we shall discuss the data mapping process, during which the data from a relational database is mapped to an object-oriented database.

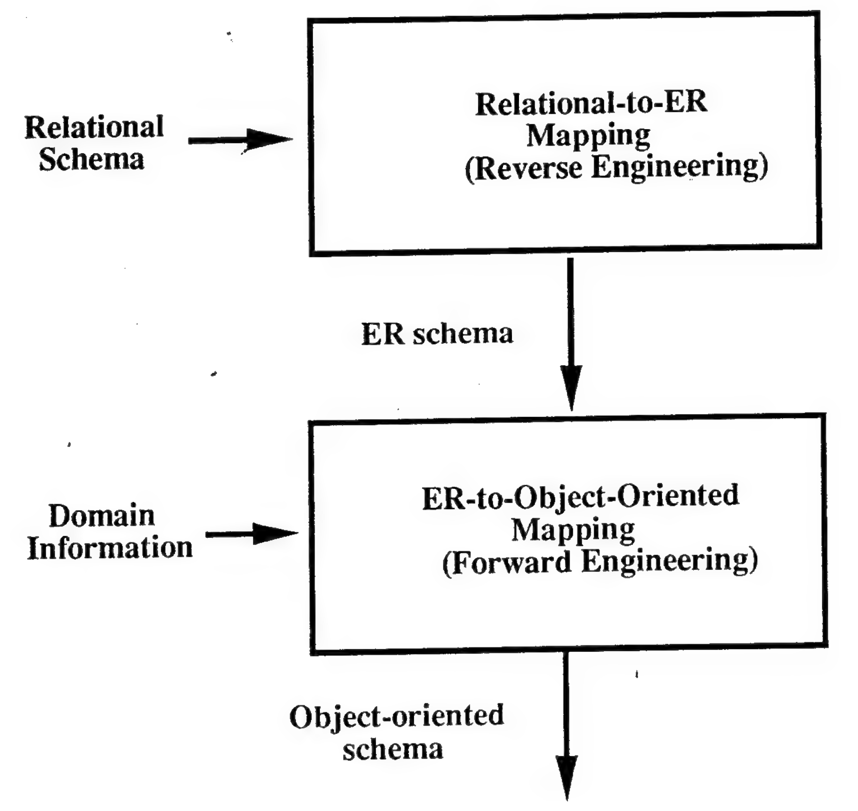
Relational Schema
Relational-to-ER Mapping (Reverse Engineering)
ER schema
Domain Information
ER-to-Object-Oriented Mapping (Forward Engineering)
Object-oriented schema

Figure 2. Mapping a Relational Schema to an Object-Oriented Schema

<u>**The Schema Mapping Process**</u>

We used an ER schema as an intermediate representation during the process of mapping from a relational schema to an object-oriented schema. This was a convenient way of familiarizing ourselves with the contents and semantics of the grid portion of the NEONS database. We found this part of the process, which is called reverse engineering, to be most helpful because it gave us a better set of questions to ask the CAST scientists when we interviewed them about the semantics of the data. Once this part of the process was done, we then mapped the resulting ER schema to an object-oriented schema using a forward engineering process.

Reverse Engineering: From a Relational Schema to an ER Schema

As stated earlier, the reverse engineering process is conceptually the reverse of a well-known technique for producing a relational schema from an ER schema (Elmasri and Navathe 1994). In the relational model, both entities and relationships are represented as relations. So we had to examine each relation to determine if it represented an entity or a relationship. Those relations which contained no foreign keys were interpreted as representing entities. The presence of foreign keys in the relations helped us to establish the many-to-many (N:M) and one-to-many (1:N) relationships among the entities.

In this process, we had to be sure that the relationships implied in the relational database schema were explicitly represented in the ER diagram. For example, the appearance of one table's primary key as a foreign key in another table implied a relationship between the two entities represented by the tables. We determined the cardinality of the relationships from the documentation available for the NEONS database (Jurkevics 1992) and from inspection of the data stored in the database (Ramanathan and Koduri 1995). When inferring the cardinality of a relationship through inspection of the data, we asked the CAST scientists to confirm our conclusion to ensure that the presence of a certain cardinality was not merely incidental.

The ER diagram was refined through consultation with the CAST scientists. It is certainly possible that more than one reasonable ER diagram could have been produced through "alternate interpretations of the structure and data" in the relational database (Premerlani and Blaha 1994). Once we felt that we had a reasonable ER diagram, we used it as a starting point for producing the object-oriented schema. From the ER diagram, we were able to define a preliminary set of object classes and their relationships.

Forward Engineering: From an ER Schema to an Object-Oriented Schema

All of the entities in the ER schema were represented as object classes in the object-oriented schema. The interpretation of the relationships in the ER schema was not as straightforward. The ER model is limited in its representation of relationships. In the ER model, a relationship is a mapping between entity types, with the cardinality of the mapping being 1:1, 1:N, or N:M. This is the extent of the semantics of the relationship that can be represented in an ER relationship. In an object-oriented model, however, there may be a number of different types of relationships. For example, there may be associations (similar to the ER concept of a relationship), aggregations (special associations that represent "a-part-of" relationships), and generalizations (superclass-subclass inheritance hierarchy relationships) (Rumbaugh et al. 1991).

We first identified generalizations. Premerlani and Blaha (1994) have suggested several clues that may be indicative of a generalization relationship in a relational schema. Some simple adaptation makes these clues also useful for ER schemas. For example, the following patterns may be generalization relationships:

- ER relationships consisting entirely of foreign keys from various entities
- entities in which many attributes from other entities have been replicated

We then identified those associations that represented a-part-of relationships. We represented those as aggregations. The ability to recognize aggregations requires "semantic understanding" (Premerlani and Blaha 1994), so we relied upon the CAST scientists for additional domain information. This information was obtained from multiple interviews with various scientists at CAST. Methods for deriving object-oriented schemas from ER diagrams cannot recognize such relationships without the benefit of domain information (Ramanathan and Koduri 1995).

By talking to scientists at CAST about the semantics of the data and how the data is used, we were able to incorporate additional semantic information into the design of the object-oriented schema. As a result, we were able to identify additional object classes for the grid data that would not have been recognized by the relational-to-ER-to-object-oriented process otherwise. A shortcoming of this part of the process is that the extraction of the additional domain information is not an automated process.

We also obtained some of the semantic information we needed about the primary grid data from the descriptive and associative realms. For example, the descriptive realm contains information about generic attribute names such as *geom_parm_1*. The interpretation of the attribute *geom_parm_1* is different for different types of grid geometry projections. The associative realm contains information about the primary data such as the numerical model that was used, the grid geometry that was used, and where the measurements were made. Thus our object-oriented approach to representing the primary grid data in the NEONS database was actually a unifying approach that made use of the appropriate information from the descriptive and associative realms as well as the primary realm. The portion of the NEONS database made available to us for this project contained two primary realm relations, two associative realm relations, and eight descriptive realm relations. The object-oriented schema produced from the relational-to-ER-to-object-oriented mapping is shown in Figure 3.

The Data Mapping Process

The data mapping process is the process of mapping the data from the NEONS database (where it is stored as relational tuples) to the object-oriented database (where it is stored as complex objects). One way of accomplishing this is by providing an object-oriented view of the existing relational data, which is what we first did as an initial investigation into the feasibility of using an object-oriented approach (Ramanathan 1994). However, this approach of providing an object-oriented wrapper around the existing relational database has two major disadvantages. First, it requires that the object-oriented view must be modified whenever the relational schema is modified. Second, it represents an additional expense at run-time due to the need to dynamically map the relational tuples to objects. Another way of accomplishing the data mapping is to actually port the data from the relational database into an object-oriented database that is managed by an object-oriented DBMS. This approach has the advantage of being a one-time operation. Unfortunately, most object-oriented DBMSs do not provide data loading routines because of the complexity of the data caused by embedded objects, relationships, inheritance hierarchies, etc. (Wiener and Naughton 1994).

We have populated the ObjNEONS database using a simple approach to the data porting problem (Ramanathan and Koduri 1995). This was done using C++ and embedded SQL. ObjectStore is tightly integrated with the C++ programming language, providing extensions for the handling of persistent data. Some of the features of ObjectStore that were used for the data porting were *inverses, collections,* and *queries. Inverses* provides a mechanism for automatically maintaining referential integrity between objects. *Collections* is a class library used to manage sets of objects. *Queries* is a feature that provides for the retrieval of objects from a collection based on some predicate (i.e., some condition that evaluates to either **true** or **false**).

We provided an application programming interface called *Grid Data Browser* that provides users with a graphical interface to the grid data (Koduri 1994; Ramanathan and Koduri 1995). The interface "resides primarily in the methods attached to the class definitions of the ObjNEONS objects" (Ramanathan and Koduri 1995). The main components of the Grid Data Browser are shown in Figure 4. Grid Data Browser was implemented using the Motif toolkit and runs on X Windows. The front-end component sends user requests to *InfoManager*, which handles the interactions with the

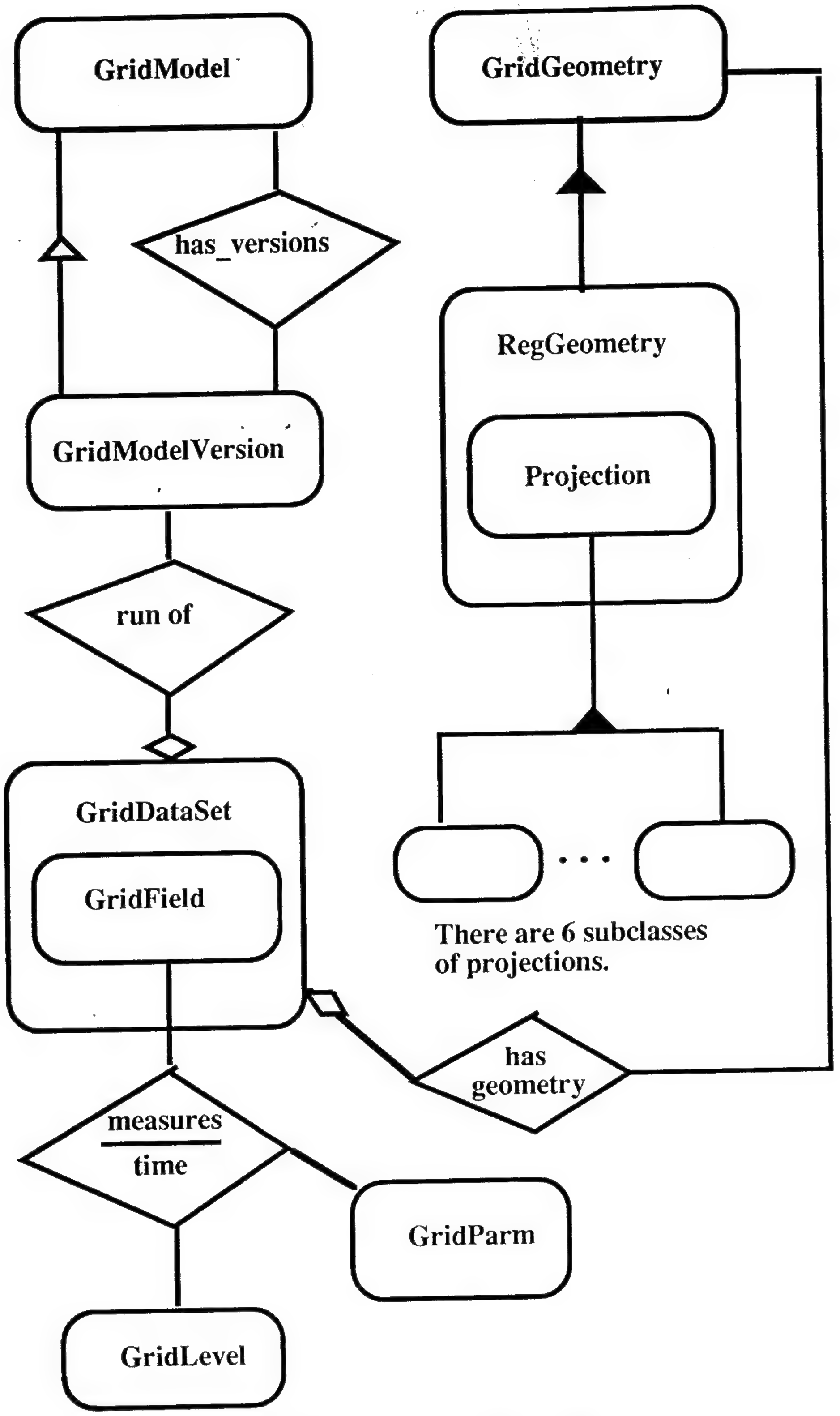

Figure 3. Object-Oriented Grid Database Diagram

ObjNEONS database. InfoManager executes the user query, then returns the result to the front-end component. All of the database operations are encapsulated in InfoManager, with the front-end of the user interface being completely decoupled from these operations. This makes it possible for the user interface to be adapted for use with other databases by modifying only the back-end component, leaving the front-end component unchanged.

Preliminary Work with Other Data Types

With the advice of the CAST scientists, we chose to work on the grid data portion of the NEONS database first. However, we have done some preliminary work on two of the other data types: the latitude-longitude-time (llt) data and the image data. Using the re-engineering process, we produced object models for the llt data and the image data, then loaded some sample data of each type into a corresponding object-oriented database. That is, we created a prototype object-oriented llt database and a prototype object-oriented image database.

The llt data consists of a set of data points representing readings taken along a line or curve. The data represents a variety of information such as conventional environmental reports, point climatology data, and earth-located satellite scene stations (Jurkevics 1992). In our initial prototype object-oriented llt database, we included only 15 of the different kinds of llt data in our object model and actually loaded the data for only three of these. We also developed a simple user interface to demonstrate that the llt data had been correctly stored and could easily be retrieved from the object-oriented database (Kalluri 1995).

Unlike the other data types stored in NEONS, image data is stored as flat files of binary data rather than being stored in relational tables. In our prototype object-oriented image database, we have stored not only satellite images, but also information about the satellites themselves (orbits, sensors, channels, etc.). Thus we have stored not only the raw images, but also descriptive information about them, in a manner similar to what we had done with the grid data. We also developed a simple interface to demonstrate that the images had been correctly stored and could easily be retrieved from the object-oriented database (Cheng 1995).

Summary and Conclusions

In this document, we have described the development of a prototype object-oriented geophysical database through the re-engineering of an existing relational database. The re-engineering process allowed us not only to map the relational schema to an object-oriented schema, but also to map the relational data to objects. We have described this process and

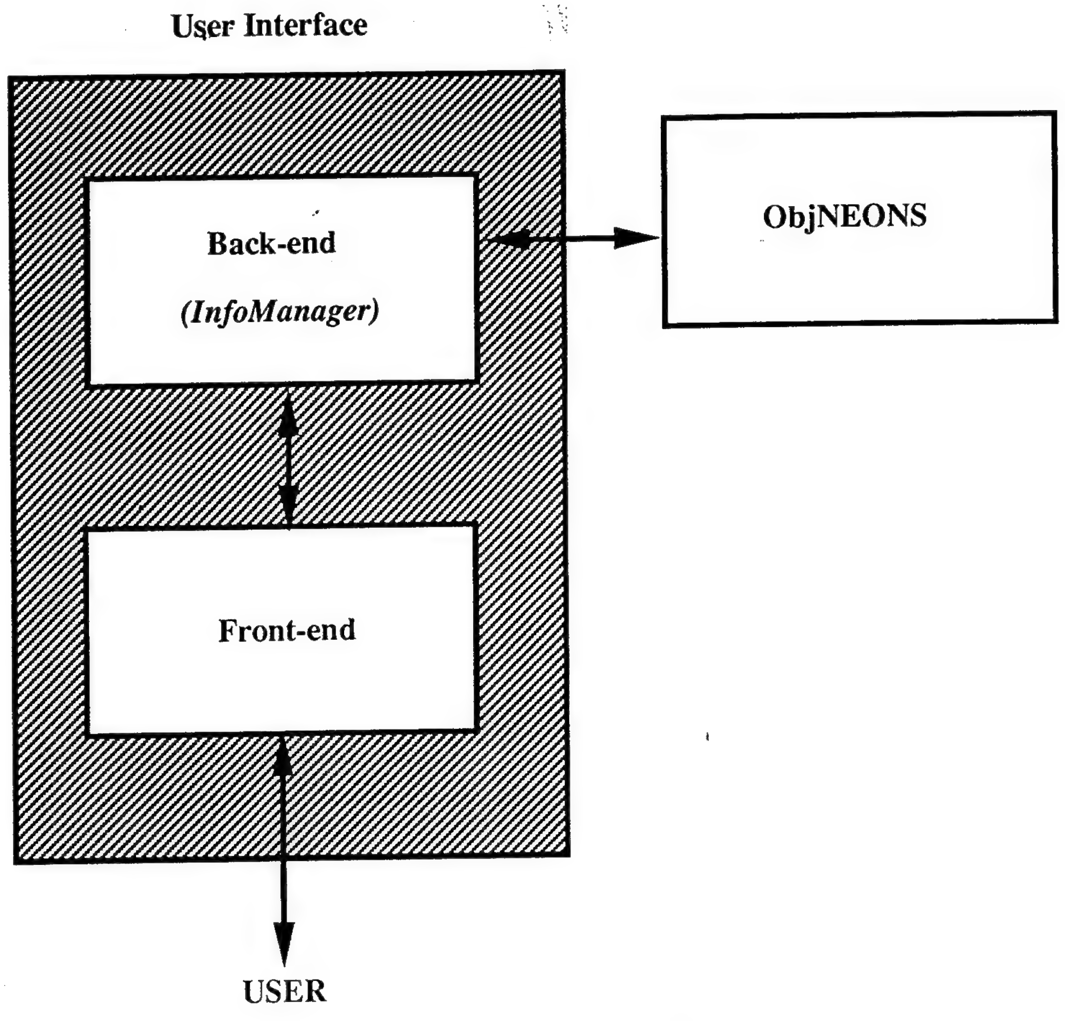

Figure 4. Grid Data Browser

its application to the grid data portion of the NEONS relational database in some detail. We have also briefly described the preliminary work that has been done in re-engineering the llt data and image data portions of NEONS.

Our re-engineering process requires that we augment the syntactic information about the structure of the relational database with semantic information about the data that is provided by domain experts. Whereas many database re-engineering efforts address only schema mapping, our process includes both schema mapping and data mapping. It is our opinion that it is important to include data mapping in this process because of the large quantities of data available in existing databases.

We intend to address ways in which the semantic information may be formalized so that the process of incorporating this information into the object-oriented design is a well-defined, systematic one rather than the informal approach used in current schema mapping efforts. We also intend to develop database loading algorithms that will allow one to port data from a relational database to an object-oriented database. Wiener and Naughton (1994) have done some work in the development of routines for the bulk loading of data into an object-oriented database. Their methods are intended to handle different types of associations, but they do not address inheritance hierarchies. Also, the source of the data for the object-oriented database is simply data files rather than data already stored in some other database such as a relational database. We plan to modify their approach so that it will handle inheritance hierarchies and be able to load an object-oriented database from a relational database.

Acknowledgments

The authors are indebted to a number of people for their contributions to this work. We are especially grateful to Mr. James Corbin, the director of the Center for Air Sea Technology (CAST), for his support and encouragement for our efforts. We have made great use of technical information that he has supplied us. We have relied upon a number of CAST scientists for advice and guidance during the development of the object-oriented schemas and the user interfaces. Most notable among these are Mr. Ramesh Krishnamagaru, Mr. Valentine Anantharaj, Ms. Cheryl Cesario, and Mr. Steve Foster.

The CAST object-oriented database project has included a number of graduate students during its duration, all of whom have made important contributions to this work. We are grateful to Dongmei Wu for her work on the user interface for the early system that provided an object-oriented view of the relational database. We are also grateful to Sridhar Koduri for the development of the Grid Data Browser as well as for some of the early work done on the design of an object-oriented schema for the image data. We are also appreciative of the contributions made by Vark Kalluri and Dawen Cheng in the areas of the llt data and the image data, respectively. All of these students attended weekly meetings for the CAST team, so that the results have truly been produced from a collaborative effort.

References

Bertino, Elisa, and Lorenzo Martino. 1991. Object-oriented database management systems: Concepts and issues. *Computer* 24 (April): 33-47.

Bhargava, Bharat. 1992. Transition from a relation to object model implementation. In *The Proceedings of the ISMM First International Conference on Information and Knowledge Management, held in Baltimore, MD, November 8-11, 1992,* by The International Society for Mini and Microcomputers - ISMM, 46-50.

Blaha, Michael R., William J. Premerlani, and James E. Rumbaugh. 1988. Relational database design using an object-oriented methodology. *Communications of the ACM* 31 (April): 414-27.

Chen, P.P. 1976. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems* 1(March): 9-36.

Cheng, Dawen. 1995. Development of an object-oriented image database system. Master's project, Mississippi State University.

Chiang, Roger, Terence Barron, and Veda Storey. 1994. Reverse engineering of relational databases: Extraction of an EER model from a relational database. *Data and Knowledge Engineering* 10 (12)" 107-42.

DeSanti, Mike, and Jeff Gomsi. 1994. A comparison of object and relational database technologies. *Object Magazine* 3 (January): 51, 56-7.

Elmasri, Ramez, and Shamkant B. Navathe. 1994. *Fundamentals of database systems.* 2d ed. Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc.

Jurkevics, Andrew. 1992. *Database design document for the Naval Environmental Operational Nowcasting System Version 3.5, June 1, 1992*, Monterey, CA: Naval Oceanographic and Atmospheric Research Laboratory.

Kalluri, Vark. 1995. Object-oriented Geophysical database for latitude, longitude, and time (llt) data. Master's project, Mississippi State University.

Koduri, Sridhar. 1994. Grid data browser: A tool to browse the data in an object-oriented database. Master's project, Mississippi State University.

Lingsch, Stephen C., and Christopher S. Robinson. 1992. Acoustic imagery using a multibeam bathymetric system. *Marine Geodesy* 15:81-95.

Markowitz, V.M., and J.A. Makowsky. 1990. Identifying extended entity-relationship object structures in relational schemas. *IEEE Transactions on Software Engineering* 16 (August): 777-90.

Markowitz, Victor M., and Arie Shoshani. 1989. On the correctness of representing extended entity-relationship structures in the relational model. In *Proceedings of the 1989 ACM SIGMOD International Conference on the Management of Data,* *Portland, Oregon, June 1989* edited by James Clifford, Bruce Lindsay, and David Maier, 430-9. New York: ACM Press.

Piatetsky-Shapiro, Gregory, and William J. Frawley, eds. 1991. *Knowledge discovery in databases*. Menlo Park, CA: AAAI Press/The MIT Press.

Premerlani, William J., and Michael R. Blaha. 1994. An approach for reverse engineering of relational databases. *Communications of the ACM* 37 (May): 42-9, 134.

Ramanathan, Chandreshekar. 1994. Providing object-oriented access to a relational database. In *Proceedings of the 32nd Annual ACM Southeast Conference, Tuscaloosa, AL, March 17-18, 1994,* edited by David W. Cordes and Susan V. Vrbsky, 162-5. New York: Association for Computing Machinery.

Ramanathan, Chandrashekar, and Julia Hodges. 1994a. An object-oriented prototype for a geophysical database subtask: Evaluation of commercial object-oriented DBMS's. In *1994 Student Research Projects*, Technical Note 04-94, September 20, 1994, 11-5, Stennis Space Center, MS: Mississippi State University Center for Air Sea Technology.

Ramanathan, Chandrashekar, and Julia Hodges. 1994b. An object-oriented prototype for a geophysical database subtask: Prototype Implementation. In *1994 Student Research Projects*, Technical Note 04-94, September 20, 1994, 19-27, Stennis Space Center, MS: Mississippi State University Center for Air Sea Technology.

Ramanathan, Chandrashekar, and Sridhar Koduri. 1995. An object-oriented prototype for a geophysical database. In *Proceedings of the Twenty-Seventh southeastern Symposium on System Theory, March 12-14, 1995, Mississippi State, Mississippi,* 170-4. Los Alamitos, CA: IEEE Computer Society Press.

Reed, Thomas Beckett IV, and Donald Hussong. 1989. Digital image processing techniques for enhancement and classification of SeaMARC II side scan sonar imagery. *Journal of Geophysical Research* 94 (B6): 7469-90.

Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. 1991. *Object-oriented modeling and design.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Wiener, Janet, and Jeffrey Naughton. 1994. *Bulk loading into an OODB: A performance study*, Technical Report TR-1218a, November 1994, Madison, Wisconsin: University of Wisconsin-Madison.

Wu, Dongmei. 1993. A window-based intelligent interface for defining descriptive data in the Navy Environmental Operational Nowcasting System (NEONS) database. In *1993 Student Research Projects,* Technical Note 02-94, December 31, 1993, 8-10, Stennis Space Center, MS: Mississippi State University Center for Air Sea Technology.

DISTRIBUTION LIST

1. Scientific Officer (Code 1232)
   Office of Naval Research
   800 N. Quincy Street
   Arlington, VA 22217-5000
   Dr. Tom Curtin (3 copies)

2. Defense Technical Information Center
   Building 5, Cameron Station
   Alexandria, VA 22304-6145
   - Two Copies

3. Defense of the Navy
   Office of Naval Research Resident
   Representative
   101 Marietta Tower-Suite 2805
   101 Marietta Street
   Atlanta, GA 30303

4. Director
   Naval Research Laboratory
   Attention: Code 2627
   Washington, DC 20375

5. Oceanographer of the Navy
   U.S. Naval Observatory
   34th and Massachusetts
   Washington, DC 20392

6. Naval Research Laboratory
   Code 7320
   Stennis Space Center, MS 39529

7. Technical Director
   Naval Oceanographic Office
   Stennis Space Center, MS 39529

8. Commanding Officer
   Fleet Numerical Meteorology
   Oceanographic Center
   Monterey, CA 93943-5000

9. Technical Director
   Naval Oceanography Command
   Building 1020
   Stennis Space Center, MS 39529

10. CDR David Markham
    Space and Naval Warfare Systems
    Command (PMW 175-3B)
    2451 Crystal Drive
    Arlington, VA 22245-5200

11. Department of Computer Science
    Mississippi State University
    Mississippi State, MS 39762
    - Dr. Julia Hodges
    - Dr. Susan Bridges
    - Mr. Shekar Ramanathan

12. Center for Air Sea Technology
    Mississippi State University
    Stennis Space Center, MS 39529
    - Mr. Jim Corbin
    - Dr. Lanny Yeske
    - Mr. Steve Foster
    - Mr. Ramesh Krishnamagaru
    - Mr. Valentine Anantharaj

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. Agency Use Only *(Leave blank).*	**2. Report Date.** 20 JUNE 1995	**3. Report Type and Dates Covered.** TECHNICAL REPORT

4. Title and Subtitle. RE-ENGINEERING A RELATIONAL DATABASE SYSTEM TO PRODUCE A PROTOTYPE OBJECT-ORIENTED GEOPHYSICAL DATABASE SYSTEM	**5. Funding Numbers.** *Program Element No.* *Project No.* *Task No.* *Accession No.*
6. Author(s). J. HODGES, S. RAMANATHAN, AND S. BRIDGES	
7. Performing Organization Name(s) and Address(es). MISSISSIPPI STATE UNIVERSITY CENTER FOR AIR SEA TECHNOLOGY STENNIS SPACE CENTER, MS 39529-6000	**8. Performing Organization Report Number.** CAST TECHNICAL REPORT 01-95
9. Sponsoring/Monitoring Agency Name(s) and Address(es). OFFICE OF NAVAL RESEARCH 800 NORTH QUINCY STREET CODE 1242 ARLINGTON, VA 22217	**10. Sponsoring/Monitoring Agency Report Number.** CAST TECHNICAL REPORT 01-95

11. Supplementary Notes.

RESEARCH PERFORMED UNDER THE OFFICE OF NAVAL RESEARCH CONTRACT/GRANT NUMBER N00014-92-J-4109

12a. Distribution/Availability Statement. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED	**12b. Distribution Code.**

13. Abstract *(Maximum 200 words).*

This document provides a description of (1) the design and development of a prototype object-oriented geophysical database system, and (2) the development of a re-engineering process for mapping from an existing relational database to an object-oriented database. Although it is widely recognized that the object-oriented approach has many advantages over traditional relational technology for scientific databases, one cannot ignore the large investments that have been made in existing relational databases. The re-engineering process described provides a systematic method for (1) mapping from an existing relational schema to an object-oriented schema, and for (2) mapping the actual data from the relational database into an object-oriented database.

14. Subject Terms. (U) Technical Report (U) Object-Oriented (U) Database (U) Relational (U) Re-Engineering	**15. Number of Pages.** 20
	16. Price Code.

17. Security Classification of Report. UNCLASSIFIED	**18. Security Classification of This Page.** UNCLASSIFIED	**19. Security Classification of Abstract.** UNCLASSIFIED	**20. Limitation of Abstract.**